Package 'rCoreGage'

May 9, 2026

Title Data Quality Check Framework for Clinical and Analytical Data

Version 1.1.0

Description A configuration-driven framework for running domain-level data quality checks and consolidating findings into structured Excel reports with role-based feedback routing.
It supports trial-level and study-level checks across multiple data domains.
Reports are routed to separate feedback channels for Data Management (DM), Medical Writing (MW), Study Data Tabulation Model (SDTM) programmers, and Analysis Data Model (ADaM) programmers, as well as other relevant data roles.
Reviewer responses are incorporated automatically on re-run.

Depends R (>= 4.1.0)

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Imports readxl (>= 1.4.0), openxlsx (>= 4.2.5), dplyr (>= 1.1.0),

Suggests haven (>= 2.5.0), testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/ganeshbabuNN/rCoreGage

BugReports https://github.com/ganeshbabuNN/rCoreGage/issues

Language en-US

NeedsCompilation no

Author Ganesh Babu G [aut, cre]

Maintainer Ganesh Babu G <ganeshbabu346@gmail.com>

Repository CRAN

Date/Publication 2026-04-25 10:00:02 UTC

Contents

`build_reports` *Consolidate findings and write role-based Excel reports*

Description

Merges new findings with previously saved issues, incorporates reviewer feedback from role-separated feedback folders, and writes four role-specific reports plus `all_open.xlsx` and `all_closed.xlsx` to `cfg$reports`.

Usage

```
build_reports(cfg, state)
```

Arguments

`cfg`	A named list of paths from `project_config.R`.
`state`	The state object returned by `run_checks`.

Details

Feedback loop: After distributing reports to reviewers, they place updated files in the appropriate subfolder under `cfg$feedback`:

- `feedback/DM/` - Data Management reviewer
- `feedback/MW/` - Medical Writing reviewer
- `feedback/SDTM/` - SDTM programmer
- `feedback/ADAM/` - ADaM programmer

On the next run, `build_reports()` reads the most recent file from each feedback folder, merges `analyst_note`, `review_note` and status updates back into the findings, and overwrites the reports with the merged version.

Value

Invisibly `NULL`. Reports are written to `cfg$reports`.

Examples

```
tmp_rep <- tempdir()
cfg <- list(
  rule_registry = system.file("extdata", "rule_registry.xlsx",
                              package = "rCoreGage"),
  trial_checks  = tmp_rep,
  study_checks  = tmp_rep,
  inputs        = tmp_rep,
  reports       = tmp_rep,
  feedback      = tmp_rep
)
state         <- setup_coregage(cfg)
state$domains <- load_inputs(cfg)
state         <- run_checks(cfg, state)
build_reports(cfg, state)
```

---

collect_findings *Collect findings from a single check into the master issues table*

Description

Called at the end of every individual check block. Validates the findings data frame, counts valid observations via count_valid, and appends to state$issues and state$summary_log.

Usage

```
collect_findings(
  state,
  df,
  id,
  desc_col = "description",
  sobs = TRUE,
  unblind_codes = character(0)
)
```

Arguments

state	The current state object.
df	A data frame produced by the check. Must contain columns: subj_id (character), vis_id (numeric), description (character, max 200 chars).
id	Character. The check ID matching the ID column in rule_registry.xlsx.
desc_col	Character. Name of the description column in df. Defaults to "description".
sobs	Logical. Whether to limit output to 20 observations. Defaults to TRUE.
unblind_codes	Character vector. Topic codes that could unblind the study. Rows with negative subject IDs where these codes are absent from the description are excluded. Defaults to character(0).

Value

Updated `state` object.

Examples

```
# Build a minimal state object
state <- list(
  issues = data.frame(id = character(0), subj_id = character(0),
                      vis_id = numeric(0), description = character(0),
                      review = character(0), stringsAsFactors = FALSE),
  summary_log = data.frame(headlink = character(0), nu = integer(0),
                           rule_set = character(0), sobs = character(0),
                           stringsAsFactors = FALSE)
)

# A findings data frame produced by a check
MY_CHECK <- data.frame(
  subj_id     = c("SUBJ-001", "SUBJ-002"),
  vis_id      = NA_real_,
  description = c("AESEV missing for RASH", "AESEV missing for HEADACHE"),
  stringsAsFactors = FALSE
)

state <- collect_findings(state, MY_CHECK, id = "AECHK001")
```

`count_valid` *Count valid observations in a findings data frame*

Description

Returns the number of rows in `df`, optionally excluding rows that could unblind the study (negative subject IDs where none of the unblinding topic codes appear in the description).

Usage

```
count_valid(df, unblind_codes = character(0))
```

Arguments

`df`	A data frame with at least `subj_id` and `description` columns.
`unblind_codes`	Character vector of topic codes that flag potential unblinding. Defaults to `character(0)` (no exclusion).

Value

Integer. Number of valid rows.

Examples

```
df <- data.frame(subj_id = c("001", "-002"),
                 description = c("Issue A", "Issue B"),
                 stringsAsFactors = FALSE)
count_valid(df)
```

`create_project` *Create a new CoreGage project*

Description

Scaffolds a complete CoreGage project folder structure at the specified path and copies all required template files from the package installation. After running this function, open the generated `.Rproj` file in RStudio and start working.

Usage

```
create_project(name, path, overwrite = FALSE)
```

Arguments

`name`	Character. Project name. Used as the folder name and `.Rproj` file name.
`path`	Character. Parent directory where the project folder will be created. Defaults to the current working directory.
`overwrite`	Logical. Whether to overwrite an existing project at the same path. Defaults to `FALSE`.

Value

Invisibly returns the full path to the created project.

Examples

```
proj_path <- file.path(tempdir(), "example_project")
dir.create(proj_path, showWarnings = FALSE)
create_project(name = "TRIAL_ABC", path = proj_path)
```

load_inputs *Load domain input data files*

Description

Reads every `.csv` and (optionally) `.sas7bdat` file from the `inputs/` folder into a named list. The name of each element is the lowercase filename without extension (e.g. `AE.csv` becomes `domains$ae`). Drop a new domain file into `inputs/` and it is picked up automatically on the next run - no code change required.

Usage

```
load_inputs(cfg)
```

Arguments

`cfg`	A named list of paths from `project_config.R`. Must contain `cfg$inputs`.

Value

A named list of data frames, one per domain file found.

Examples

```
# Create a temporary inputs folder with a sample CSV
tmp_inp <- tempdir()
write.csv(
  data.frame(USUBJID = "SUBJ-001", AETERM = "RASH"),
  file.path(tmp_inp, "AE.csv"), row.names = FALSE
)
cfg     <- list(inputs = tmp_inp)
domains <- load_inputs(cfg)
# domains$ae contains the loaded Adverse Events data
```

run_checks *Execute all active check programs*

Description

Loops through all active rule sets defined in the rule registry, sources each check script from the appropriate folder (trial checks from `cfg$trial_checks`, study checks from `cfg$study_checks`), and calls `check_RULESET(state, cfg)`, where `RULESET` is the value from the `Rule_Set` column in `rule_registry.xlsx`.

Usage

```
run_checks(cfg, state)
```

Arguments

cfg	A named list of paths from project_config.R.
state	The state object returned by setup_coregage.

Value

Updated state object with issues and summary_log populated.

Examples

```
tmp_rep <- tempdir()
cfg <- list(
  rule_registry = system.file("extdata", "rule_registry.xlsx",
                              package = "rCoreGage"),
  trial_checks  = tmp_rep,
  study_checks  = tmp_rep,
  inputs        = tmp_rep,
  reports       = tmp_rep,
  feedback      = tmp_rep
)
state          <- setup_coregage(cfg)
state$domains  <- load_inputs(cfg)
state          <- run_checks(cfg, state)
```

---

setup_coregage *Initialise a CoreGage run session*

Description

Reads the rule registry (rule_registry.xlsx), builds the active rule switch vector, and initialises all empty master tables required for a check run. This is the first function called in every run.

Usage

```
setup_coregage(cfg)
```

Arguments

cfg	A named list of paths produced by sourcing project_config.R. Must contain: rule_registry, reports, feedback.

Value

A named list (the *state* object) containing: `rule_registry`, `active_rules`, `session`, `issues`, `summary_log`, `review_log`.

Examples

```
tmp_rep <- tempdir()
cfg <- list(
  rule_registry = system.file("extdata", "rule_registry.xlsx",
                              package = "rCoreGage"),
  trial_checks  = tmp_rep,
  study_checks  = tmp_rep,
  inputs        = tmp_rep,
  reports       = tmp_rep,
  feedback      = tmp_rep
)
state <- setup_coregage(cfg)
```

Index